Cryptography

Crypto Project – 23 VI

Each Project: 50 points. You need both: present an implementation and provide a short report describing: problem statement, description of the cryptographic building blocks and proposed solution/results.
Each project can be prepared by up to 2 students.
Projects for Algorithmic track are marked with A. Projects for Security track are marked with S.

1. (S) Implement an attack described in the [4].
2. (S) Modify TLS library in such a way that the random values used in TLS handshake are predictable to you but are unpredictable to anyone else (kleptography). Then write a script which given a (*e.g.,* use tdpdump) transcript of the TLS session decrypts it for you. Use the combination wireshark and firefox to automate.
3. (A, S) Implement dining-cryptographers protocol.
4. (A, S) Implement a mobile app which has a functionality similar to "Tinderella" (`www.youtube.com/watch?v=bLoRPielarA`). The goal of the protocol performed between users P_i and P_2 with inputs x_1, x_2 respectively is to compute $F_i(x_i, x_2) = F_2(x_1, x_2) = b$ where $b = x_1 \wedge x_2$.

   Your app should not let anyone to learn about user's inputs (you can use Yao's garbled circuits or GSW protocol).
5. (A, S) Write a mobile app which lets two users to tell if they are nearby. Implemented functionality returns 1 if two users are within a given distance and returns 0 in any other case.
6. (S) Implement a command-line Helios voting [1] client.

   Your implementation should output:

   - an encryption of a ballot,
   - (together with) zero-knowledge proof of knowledge of exponent (randomness used).
   - commitment to randomness.

   Follow technical description which you can find on the website [2]. The output of your program needs to be in json format and be compatible with the Helios implementation *i.e.,* one should be able to use Helios verifier (`https://github.com/benadida/helios-server/blob/master/heliosverifier/verify.html`) to audit correctness of your ballot encryption.
7. (S) Implement a mobile app which lets one to verify correctness of ecnrypions in Helios voting system. The verifier takes as an input an encrypted ballot (in json format) and checks correctness of zero-knowledge proofs attached to it.
8. (A, S) Prepare a mobile app(s) which implements Feige-Fiat-Shamir identification protocol. Such an app can be used then as user's digital ID.
9. (A, S) Implement problem 3 from Lab 2.
10. (A, S) Implement problem C from Lab 4.
11. (A, S) Implement 2-out-of-n threshold encryption for an app which stores an encrypted data backup in a cloud. In order to decrypt data one needs to use two devices (*e.g.,* PC + phone).

12. (A) Choose an NP-hard problem and use it as a signature schme (start with finding a zero-knowledge proof for it and then apply Fiat-Shamir heuristic).

13. (A, S) Two financial institutions have separate databases of customers. Law does not allow them to reveal information about each customer (and banks are not willing to reveal to each other this information). But they are allowed (and willing) to share information about people who are at the same time the customers of both banks.

    Ron is blacklisted in the Gringotts Wizzarding Bank and he asks for a loan in Uncle Vernon's Bank. How Gringotts can warn about Ron not knowing if Ron is Uncle's customer?

    Design and implement a protocol which allows for computing the intersection of two data sets. The protocol should not reveal any other information about the contents of the sets.

14. (A, S) Implement 2-party lottery, use bitcoin to guarantee honesty of the participants [3].

References

[1] Ben Adida. Helios: Web-based open-audit voting. In *USENIX Security Symposium*, volume 17, pages 335–348, 2008.

[2] Ben Adida. http://documentation.heliosvoting.org/verification-specs/helios-v4, 2012.

[3] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. Fair two-party computations via bitcoin deposits. In *International Conference on Financial Cryptography and Data Security*, pages 105–121. Springer, 2014.

[4] Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient cache attacks on aes, and countermeasures. *Journal of Cryptology*, 23(1):37–71, 2010.